WHAT IS CLAIMED IS:

1. A method for managing a virtual heap for a process executing within a virtual machine executing within a device, the method comprising:

providing a store heap for the process, wherein the store heap is comprised in the virtual heap;

providing an in-memory heap for the process, wherein the in-memory heap comprises a cached portion of the store heap for the process, and wherein the in-memory heap is comprised in the virtual heap;

performing an atomic transaction on the virtual heap, wherein said performing the atomic transaction comprises performing one or more transaction tasks, and wherein said performing the atomic transaction changes a state of the virtual heap by modifying one or more portions of the virtual heap;

committing the atomic transaction by accepting the modifications to the one or more portions of the virtual heap if the one or more transaction tasks in the atomic transaction are performed without generating an error; and

rejecting the atomic transaction by restoring the virtual heap to the state of the virtual heap prior to said performing the atomic transaction if one or more of the one or more transaction tasks in the atomic transaction generates an error when performed.

2. The method of claim 1,

wherein an access state of the store heap is closed prior to said performing the atomic transaction, and wherein the closed access state prohibits performing the atomic transaction;

the method further comprising:

changing the access state of the store heap to open prior to said performing the atomic transaction, wherein the open access state permits said performing the atomic transaction.

3. The method of claim 2, further comprising:

changing the access state of the store heap to closed subsequent to said performing the atomic transaction.

4. The method of claim 1,

wherein the atomic transaction is an atomic write transaction; and

wherein said performing the atomic transaction comprises:

reading a first portion of the in-memory heap; and

writing the first portion of the in-memory heap to the store heap.

5. The method of claim 4,

wherein said performing the atomic transaction further comprises:

verifying that the first portion of the in-memory heap is successfully read from the in-memory heap prior to said writing the first portion of the in-memory heap to the store heap.

6. The method of claim 4,

wherein said performing the atomic transaction further comprises:

deleting the first portion from the in-memory heap subsequent to said reading the first portion from the in-memory heap.

7. The method of claim 1,

wherein the atomic transaction is an atomic read transaction; and

wherein said performing the atomic transaction comprises:

reading a second portion of the store heap; and

writing the second portion of the store heap to the in-memory heap.

8. The method of claim 7,

wherein said performing the atomic transaction further comprises:

verifying that the second portion of the store heap is successfully read from the store heap prior to said writing the first portion of the store heap to the in-memory heap.

9. The method of claim 1,

wherein the atomic transaction is an atomic delete transaction; and

wherein said performing the atomic transaction comprises:

deleting a third portion of the store heap.

10. The method of claim 1, further comprising:

checkpointing the store heap to a persistent store to make the virtual heap persistent.

11. The method of claim 1,

wherein the store heap is one of a plurality of store heaps in a persistent store;

wherein each of the plurality of store heaps is associated with one of a plurality of processes; and

wherein the process is one of the plurality of processes.

12. The method of claim 1,

wherein the store heap and the in-memory heap are comprised in one memory address space.

13. The method of claim 1,

wherein the device is a mobile computing device.

14. The method of claim 1,

wherein the virtual machine is a Java virtual machine; and wherein the process is a Java application.

15. The method of claim 1,

wherein the in-memory heap and the store heap comprise objects for the process, and wherein the objects comprise code and data for use by the process during execution within the virtual machine.

16. A method for managing a virtual heap on a virtual machine executing within a device, the method comprising:

providing a store heap for a first process executing within the virtual machine, wherein the store heap is comprised in the virtual heap;

providing an in-memory heap for the first process, wherein the in-memory heap comprises a cached portion of the store heap for the first process, and wherein the in-memory heap is comprised in the virtual heap;

providing an application programming interface (API) for performing heap operations on the virtual heap, wherein the API comprises functions for performing operations on portions of the virtual heap, and wherein the functions in the API are callable by processes executing within the virtual machine;

a second process calling a first function from the API to perform a first operation on a first portion of the virtual heap;

performing the first operation on the first portion of the virtual heap in response to the second process calling the first function, wherein said performing the first operation changes a state of the virtual heap by modifying the first portion of the virtual heap;

committing the first operation on the first portion of the virtual heap by accepting the modifications to the first portion of the virtual heap if the first operation is performed without generating an error; and

rejecting the first operation on the first portion of the virtual heap by restoring the virtual heap to the state of the virtual heap prior to said performing the first operation if the first operation generates an error when performed.

17. The method of claim 16, wherein the first process and the second process are the same process.

18. The method of claim 16, wherein the second process is a heap management process.

19. The method of claim 16,

wherein the first operation performed by the first function called by the second process is an atomic write transaction; and

wherein the atomic transaction comprises:

reading a first portion of the in-memory heap; and

writing the first portion of the in-memory heap to the store heap.

20. The method of claim 19,

wherein the atomic transaction further comprises:

verifying that the first portion of the in-memory heap is successfully read from the in-memory heap prior to said writing the first portion of the in-memory heap to the store heap.

21. The method of claim 19,

wherein the atomic transaction further comprises:

deleting the first portion from the in-memory heap subsequent to said reading the first portion from the in-memory heap.

22. The method of claim 16,

wherein the first operation performed by the first function called by the second process is an atomic read transaction; and

wherein the atomic transaction comprises:

reading a second portion of the store heap; and

writing the second portion of the store heap to the in-memory heap.

23. The method of claim 22,

wherein the atomic transaction further comprises:

verifying that the second portion of the store heap is successfully read from the store heap prior to said writing the first portion of the store heap to the in-memory heap.

24. The method of claim 16,

wherein the first operation performed by the first function called by the second process is an atomic delete transaction; and

wherein said performing the atomic transaction comprises:

deleting a third portion of the store heap.

25. The method of claim 16, further comprising:

checkpointing the store heap to a persistent store to make the virtual heap persistent.

26. The method of claim 16,

wherein the store heap is one of a plurality of store heaps in a persistent store;

wherein each of the plurality of store heaps is associated with one of a plurality of processes; and

wherein the process is one of the plurality of processes.

27. The method of claim 16,

wherein the store heap and the in-memory heap are comprised in one memory address space.

28. The method of claim 16,

wherein the device is a mobile computing device.

29. The method of claim 16,

wherein the virtual machine is a Java virtual machine; and wherein the process is a Java application.

30. The method of claim 16,

wherein the in-memory heap and the store heap comprise objects for the process, and wherein the objects comprise code and data for use by the process during execution within the virtual machine.

31. A system comprising:

a device configured to execute a virtual machine, wherein the virtual machine is configured to execute a process;

a first memory coupled to the device, wherein the first memory is configured to store a store heap for the process, wherein the store heap is comprised within a virtual heap for the process;

a second memory coupled to the device, wherein the second memory is configured to store an in-memory heap for the process, wherein the in-memory heap is comprised within the virtual heap, wherein the in-memory heap comprises cached portions of the store heap for access by the process;

wherein the device is configured to perform operations on the virtual heap according to an application programming interface (API), and wherein the API comprises functions for performing the operations on the virtual heap, and wherein the functions in the API are callable by the process, and wherein the API is configured to:

perform an atomic transaction on the virtual heap, wherein the atomic transaction comprises one or more transaction tasks, and wherein the atomic transaction changes a state of the virtual heap by modifying one or more portions of the virtual heap;

commit the atomic transaction by accepting the modifications to the one or more portions of the virtual heap if the one or more transaction tasks in the atomic transaction are performed without generating an error; and

reject the atomic transaction by restoring the virtual heap to the state of the virtual heap prior to the atomic transaction if one or more of the one or more transaction tasks in the atomic transaction generates an error when performed.

32. The system of claim 31,

wherein an access state of the store heap is closed prior to said performing the atomic transaction, wherein the closed access state prohibits performing the atomic transaction; and

wherein the API is further configured to:

change the access state of the store heap to open prior to said performing the atomic transaction, wherein the open access state permits said performing the atomic transaction; and

change the access state of the store heap to closed subsequent to said performing the atomic transaction.

33. The system of claim 31,

wherein the atomic transaction is an atomic write transaction; and

wherein, in performing the atomic transaction, the API is further configured to:

read a first portion of the in-memory heap; and

write the first portion of the in-memory heap to the store heap.

34. The system of claim 33,

wherein, in performing the atomic transaction, the API is further configured to:

verify that the first portion of the in-memory heap is successfully read from the in-memory heap prior to said writing the first portion of the in-memory heap to the store heap.

35. The system of claim 33,

wherein, in performing the atomic transaction, the API is further configured to:

delete the first portion from the in-memory heap subsequent to said reading the first portion from the in-memory heap.

36. The system of claim 31,

wherein the atomic transaction is an atomic read transaction; and

wherein, in performing the atomic transaction, the API is further configured to:

read a second portion of the store heap; and

write the second portion of the store heap to the in-memory heap.

37. The system of claim 36,

wherein, in performing the atomic transaction, the API is further configured to:

verify that the second portion of the store heap is successfully read from the store heap prior to said writing the first portion of the store heap to the in-memory heap.

38. The system of claim 31,

wherein the atomic transaction is an atomic delete transaction; and

wherein, in performing the atomic transaction, the API is further configured to:

delete a third portion of the store heap.

39. The system of claim 31,

wherein the first memory further comprises a persistent store configured to store a plurality of store heaps, wherein the store heap for the process is one of the plurality of store heaps; and

wherein the device is further configured to execute heap management software for managing the virtual heap, and wherein the heap management software is configured to:

checkpoint the store heap for the process to the persistent store to make the virtual heap for the process persistent.

40. The system of claim 31,

wherein the store heap and the in-memory heap are comprised in one memory address space.

41. The system of claim 31,

wherein the device is a mobile computing device.

42. The system of claim 31,

wherein the virtual machine is a Java virtual machine; and wherein the process is a Java application.

43. The system of claim 31,

wherein the in-memory heap and the store heap comprise objects for the process, and wherein the objects comprise code and data for use by the process during execution within the virtual machine.

44. A carrier medium comprising programming instructions executable to manage a virtual heap for a process executing within a virtual machine executing within a device, wherein the program instructions are executable to implement:

providing a store heap for the process, wherein the store heap is comprised in the virtual heap;

providing an in-memory heap for the process, wherein the in-memory heap comprises a cached portion of the store heap for the process, and wherein the in-memory heap is comprised in the virtual heap;

performing an atomic transaction on the virtual heap, wherein said performing the atomic transaction comprises performing one or more transaction tasks, and wherein said performing the atomic transaction changes a state of the virtual heap by modifying portions of the virtual heap;

committing the atomic transaction by accepting the modifications to the portions of the virtual heap if the one or more transaction tasks in the atomic transaction are performed without generating an error; and

rejecting the atomic transaction by restoring the virtual heap to the state of the virtual heap prior to said performing the atomic transaction if one or more of the one or more transaction tasks in the atomic transaction generates an error when performed.

45. The carrier medium of claim 44,

wherein an access state of the store heap is closed prior to said performing the atomic transaction, and wherein the closed access state prohibits performing the atomic transaction;

wherein the program instructions are further executable to implement:

changing the access state of the store heap to open prior to said performing the atomic transaction, wherein the open access state permits said performing the atomic transaction; and

changing the access state of the store heap to closed subsequent to said performing the atomic transaction.

46. The carrier medium of claim 44,

wherein the atomic transaction is an atomic write transaction; and

wherein the program instructions are further executable to implement:

reading a first portion of the in-memory heap; and

writing the first portion of the in-memory heap to the store heap.

47. The carrier medium of claim 46,

wherein, in said performing the atomic transaction, the program instructions are further executable to implement:

verifying that the first portion of the in-memory heap is successfully read from the in-memory heap prior to said writing the first portion of the in-memory heap to the store heap.

48. The carrier medium of claim 46,

wherein, in said performing the atomic transaction, the program instructions are further executable to implement:

deleting the first portion from the in-memory heap subsequent to said reading the first portion from the in-memory heap.

49. The carrier medium of claim 44,

wherein the atomic transaction is an atomic read transaction; and

wherein, in said performing the atomic transaction, the program instructions are further executable to implement:

reading a second portion of the store heap; and

writing the second portion of the store heap to the in-memory heap.

50. The carrier medium of claim 49,

wherein, in said performing the atomic transaction, the program instructions are further executable to implement:

verifying that the second portion of the store heap is successfully read from the store heap prior to said writing the first portion of the store heap to the in-memory heap.

51. The carrier medium of claim 44,

wherein the atomic transaction is an atomic delete transaction; and

wherein, in said performing the atomic transaction, the program instructions are further executable to implement:

deleting a third portion of the store heap.

52. The carrier medium of claim 44,

wherein the store heap is one of a plurality of store heaps in a persistent store;

wherein each of the plurality of store heaps is associated with one of a plurality of processes;

wherein the process is one of the plurality of processes; and

wherein the program instructions are further executable to implement:

checkpointing the store heap to the persistent store to make the virtual heap persistent.

53. The carrier medium of claim 44,

wherein the store heap and the in-memory heap are comprised in one memory address space.

54. The carrier medium of claim 44,

wherein the device is a mobile computing device.

55. The carrier medium of claim 44,

wherein the virtual machine is a Java virtual machine; and wherein the process is a Java application.

56. The carrier medium of claim 44,

wherein the in-memory heap and the store heap comprise objects for the process, and wherein the objects comprise code and data for use by the process during execution within the virtual machine.